Homework 5 NT

August 5, 2023

1 Homework 5

1.0.1 Nelson Tran

1.0.2 8/5/23

Answer each question by writing the Python code needed to perform the task. Please only use the libraries requested in each problem.

1.0.3 Problem 1

Load the interest_inflation data from the statsmodels library as a pandas data frame assigned to `df`. Use the function `df.head()` to view the first 5 rows of the data. Notice the first observation is indexed at 0. Unlike R, Python is a 0 based index language which means when you iterate or wish to view the first observation of a data object it will be at the index 0.

What do the columns `Dp` and `R` represent? (You can find this using the documentation)

[23]:
```
# your code here
from statsmodels.datasets.interest_inflation.data import load_pandas
df=load_pandas().data
df.head()

#Dp represents "Delta log gdp deflator"
#R represents "nominal long term interest rate"
```

[23]:

	year	quarter	Dp	R
0	1972.0	2.0	-0.003133	0.083
1	1972.0	3.0	0.018871	0.083
2	1972.0	4.0	0.024804	0.087
3	1973.0	1.0	0.016278	0.087
4	1973.0	2.0	0.000290	0.102

1.0.4 Problem 2

Import scipy as sp and numpy as np. Using the `mean()` and `var()` function from scipy, validate that both functions equate to their numpy counterparts against the column `Dp`.

By using the scipy library you should receive a warning message. What does the warning message indicate? Which function should you use going forward?

[24]:
```
# your code here
import scipy as sp
import numpy as np

np.mean(df)
np.var(df)
sp.mean(df)
sp.var(df)

#Warning message reads that sp.mean and sp.var will be removed in version 2.0.0.␣
 ↪We will use numpy going forward
```

```
C:\Users\nelso_a7ain06\AppData\Local\Temp\ipykernel_9464\200561093.py:7:
DeprecationWarning: scipy.mean is deprecated and will be removed in SciPy 2.0.0,
use numpy.mean instead
  sp.mean(df)
C:\Users\nelso_a7ain06\AppData\Local\Temp\ipykernel_9464\200561093.py:8:
DeprecationWarning: scipy.var is deprecated and will be removed in SciPy 2.0.0,
use numpy.var instead
  sp.var(df)
```

[24]:
```
year       59.639444
quarter     1.240458
Dp          0.000353
R           0.000270
dtype: float64
```

1.0.5 Problem 3

Fit an OLS regression (linear regression) using the statsmodels api where `y = df['Dp']` and `x = df['R']`. By default OLS estimates the theoretical mean of the dependent variable y. Statsmodels.ols does not fit a constant value by default so be sure to add a constant to `x`. Extract the coefficients into a variable named `res1_coefs`. See the documentation for `params`. Finally print the `summary()` of the model.

Documentation: https://www.statsmodels.org/dev/generated/statsmodels.regression.linear_model.OLS.html

[25]:
```
# your code here
import statsmodels.api as sm
import numpy as np
y = df['Dp']
x = sm.add_constant(df['R'])
model = sm.OLS(y,x)
result = model.fit()
res1_coefs = result.params
print(result.summary())
```

```
                            OLS Regression Results
==============================================================================
```

```
Dep. Variable:                     Dp   R-squared:                       0.018
Model:                            OLS   Adj. R-squared:                  0.009
Method:                 Least Squares   F-statistic:                     1.954
Date:                Sat, 05 Aug 2023   Prob (F-statistic):              0.165
Time:                        10:06:46   Log-Likelihood:                274.44
No. Observations:                 107   AIC:                            -544.9
Df Residuals:                     105   BIC:                            -539.5
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0031      0.008     -0.370      0.712      -0.020       0.014
R              0.1545      0.111      1.398      0.165      -0.065       0.374
==============================================================================
Omnibus:                       11.018   Durbin-Watson:                   2.552
Prob(Omnibus):                  0.004   Jarque-Bera (JB):                3.844
Skew:                          -0.050   Prob(JB):                        0.146
Kurtosis:                       2.077   Cond. No.                         61.2
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

1.0.6 Probelm 4

Fit a quantile regression model using the statsmodels api using the formula `Dp ~ R`. By default quantreg creates a constant so there is no need to add one to this model. In your `fit()` method be sure to set `q = 0.5` so that we are estimating the theoritical median. Extract the coefficients into a variable named `res2_coefs`. Finally print the `summary()` of the model.

Documentation: https://www.statsmodels.org/dev/generated/statsmodels.regression.quantile_regression.QuantRe

[26]:
```
# your code here
import statsmodels.api as sm
model = sm.QuantReg(y,x)
result = model.fit(q = 0.5)
res2_coefs = result.params
print(result.summary())
```

```
                         QuantReg Regression Results
==============================================================================
Dep. Variable:                     Dp   Pseudo R-squared:               0.02100
Model:                       QuantReg   Bandwidth:                      0.02021
Method:                 Least Squares   Sparsity:                       0.05748
Date:                Sat, 05 Aug 2023   No. Observations:                  107
Time:                        10:06:50   Df Residuals:                      105
                                        Df Model:                            1
```

```
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0054      0.013     -0.417      0.677      -0.031       0.020
R              0.1818      0.169      1.075      0.285      -0.153       0.517
==============================================================================
```

1.0.7 Problem 5

Part 1: Use the `type()` method to determine the type of `res1_coefs` and `res2_coefs`. Print the type in a Jupyter cell.

Part 2: In the next Jupyter cell show that `res1_coefs > res2_coefs`. What does the error mean? To resolve this error we must convert the data to an unnamed object or change the names of the objects. Since we are not focusing on pandas this week we will simply convert to a different data type.

Part 3: Now, do the same comparision using the `tolist()` function at the end of each object name.

Part 4: We performed two types of linear regression and compared their coefficients. Coefficients are essentially the rate at which x changes the values of y. Do some research on what OLS estimates versus what quantreg estimates and explain why we have two different coefficient estimates. In which cases do you think quantile regression will be useful? What about ordinary least squares regression?

[27]:
```
# your code here
print(type(res1_coefs))
print(type(res2_coefs))
```

```
<class 'pandas.core.series.Series'>
<class 'pandas.core.series.Series'>
```

[31]:
```
print(res1_coefs > res2_coefs)
#I'm not getting an error here for the output, what type of error am I supposed␣
 ↪to be getting here?
```

```
const     True
R        False
dtype: bool
```

[32]:
```
print(res1_coefs.tolist() > res2_coefs.tolist())
```

```
True
```

OLS estimates mean vs. Quantreg estimates the median. I believe the reason why we have two different coefficient estimates is because OLS mean estimates is more affected by outliers in data. Quantile regression could be used models where there is little to no correlation between variables and quantile regression can help be a predictive model. OLS regression would be more useful when OLS conditions are met, and the researcher is more interested in the mean response of how a dependent variable changes with an independent variable.